Plan Técnico de Pruebas de Seguridad

Introducción

Propósito: Definir metodología, casos de prueba, procedimientos y criterios de aceptación para evaluar la seguridad y la resiliencia del sistema bajo distintos escenarios, incluyendo pentesting interno, rendimiento bajo ataque, análisis de vulnerabilidades y verificación de configuración segura.

1. Alcance y Suposiciones

Alcance:

- Aplicación web (CMS WordPress/Drupal) desplegada en Docker Swarm con múltiples réplicas.
- Capa de balanceo y terminación TLS en HAProxy con Keepalived (VIP activo/pasivo).
- Almacenamiento compartido GlusterFS para media/uploads.
- Base de datos MariaDB (replicación asincrónica o clúster Galera).
- Red y firewalls relacionados, políticas de seguridad, secretos y configuración de contenedores.

Suposiciones:

- Entorno de pruebas (staging) con paridad de configuración respecto a producción.
- Accesos consensuados (reglas de enganche) y ventanas de pruebas autorizadas.
- Usuarios de prueba y datos sintéticos disponibles para ejecuciones controladas.

2. Reglas de Enganche (Rules of Engagement)

- Tipos de pruebas autorizadas: DAST no destructivo, pruebas de estrés controladas, escaneo autenticado en hosts de pruebas.

- No ejecutar pruebas de denegación de servicio en producción. Usar staging o ventanas de baja actividad.

- Límites de carga: acordar TPS/usuarios máximos y duración con Operaciones.

- Reporte inmediato de hallazgos críticos (RCE, auth bypass, credenciales expuestas).

3. Metodología y Marco de Referencia

Referencias: OWASP Testing Guide v5, OWASP ASVS, NIST SP 800-115, CIS Docker Benchmark, CVSS v3.1.

Enfoque: combinar pruebas dinámicas (DAST) con auditorías de configuración y escaneo de vulnerabilidades, priorizando hallazgos por probabilidad e impacto.

4. Matriz de Riesgo y Criterios de Aceptación

Severidad (CVSS v3.1):

- Crítica (9.0–10.0): Debe mitigarse antes del go-live.
- Alta (7.0–8.9): Mitigar o compensar antes del go-live.
- Media (4.0–6.9): Plan de remediación en 30 días.
- Baja (0.1–3.9): Mitigar progresivamente.

Criterios de aceptación:

- 0 vulnerabilidades Críticas y Altas sin mitigación.
- Cabeceras de seguridad y TLS conforme a mejores prácticas.
- Resiliencia demostrada bajo escenarios de carga hostil definidos.

5. Inventario y Superficies de Ataque

Componentes:

- HAProxy/Keepalived (VIP, certificados TLS, ACLs, rate limiting/stick-tables).
- Servicios web (NGINX/PHP-FPM o apache+php-fpm), réplicas en Swarm.
- CMS (WordPress/Drupal), plugins/módulos/temas.
- MariaDB (InnoDB/Galera), usuarios/grants, parámetros.
- GlusterFS (montajes, permisos, latencia, split-brain safeguards).
- Red: overlay Swarm, firewalls, puertos expuestos.

6. Pentesting Interno (OWASP ZAP, Nikto, sqlmap)

6.1 Preparación

- Crear usuarios de prueba con roles diferenciados (admin/editor/author/reader).
- Mapear rutas críticas (login, CRUD de contenido, uploads, endpoints API).
- Habilitar proxy en navegador para ZAP (127.0.0.1:8080).

6.2 OWASP ZAP – Baseline y Active Scan

Comandos sugeridos (Docker):

1) Baseline (no intrusivo):

docker run -t owasp/zap2docker-stable zap-baseline.py -t https://sitio-ejemplo.com -r zap_baseline.html --auto

2) Full/Active scan (intrusivo, usar en staging):

docker run -t owasp/zap2docker-stable zap-full-scan.py -t https://staging.sitio.com -r zap_full.html -z "-config scanner.threadPerHost=5 -addonupdate"

Cobertura recomendada:

- Autenticación: configurar Context y usuarios en ZAP para cubrir áreas protegidas.
- Session management: verificar cookies con atributos Secure/HttpOnly/SameSite.
- Cabeceras: CSP, HSTS, X-Frame-Options, X-Content-Type-Options, Referrer-Policy.
- Vulnerabilidades: XSS, SQLi, LFI/RFI, SSRF, Directory Traversal, IDOR.

6.3 Nikto – Enumeración de servidor web

Ejemplos:

nikto -host https://staging.sitio.com -Tuning 123bde -Display V -o nikto_report.html -Format html

nikto -host https://staging.sitio.com -Plugins headers,ssl -ssl

Validar: banners, métodos inseguros, archivos por defecto, ciphers débiles, TLS/SSL.

6.4 sqlmap – Inyección SQL

Casos típicos y comandos:

Parámetros GET:

sqlmap -u "https://staging.sitio.com/item.php?id=1" --batch --risk=3 --level=5 --thread=5 --dbms=MariaDB --random-agent --tamper=space2comment

Formularios POST (capturar con Burp/ZAP y guardar request):

sqlmap -r login_request.txt --batch --risk=3 --level=5 --smart --flush-session

Cookies/Headers:

sqlmap -u "https://staging.sitio.com/" --cookie="PHPSESSID=..." --headers="X-Forwarded-For: 127.0.0.1" --batch

Objetivos: confirmar explotación, identificar tablas sensibles, verificar protecciones (WAF, parametrización, roles DB).

7. Pruebas de Rendimiento bajo Ataque (Apache JMeter)

Escenarios recomendados:

- Baseline: 100–200 usuarios, ramp-up 5–10 min, duración 20 min.
- Spike: incrementar súbitamente usuarios (x3) por 5 min y observar resiliencia.
- Stress: aumentar usuarios hasta alcanzar 95% de CPU en web pods.
- Soak/Resistencia: 2–4 horas con carga sostenida.

Métricas clave: Tiempo medio (T50/T95/T99), Throughput (req/s), error rate, saturación CPU/RAM en contenedores, latencia en HAProxy, errores 5xx.

Buenas prácticas JMeter:

- Usar HTTP Cache Manager para simular navegadores reales.
- HTTP Header Manager con User-Agent y cookies.
- Concurrency Thread Group (Plugins) para modelar usuarios reales.
- Think Time (Uniform Random Timer) para variabilidad.
- Correlación de tokens/CSRF con RegEx Extractor.

Ejecución no-GUI:

```
jmeter -n -t escenario_ataque.jmx -l resultados.jtl -e -o ./reporte
```

8. Análisis de Vulnerabilidades (OpenVAS / Nessus)

Alcance:

- Rango de red de hosts de Swarm, balanceadores y DB.
- Escaneos autenticados (SSH) en nodos Linux para mayor cobertura.

Procedimiento:

1) Configurar políticas de escaneo completas y credenciales.
2) Excluir IPs de producción sensibles si el riesgo es alto.
3) Programar ventanas de escaneo y limitar tasa de paquetes.
4) Exportar reportes y mapear CVEs a imágenes/paquetes.

Criterios de aceptación: 0 vulnerabilidades críticas abiertas; vulnerabilidades altas con plan de remediación aprobado.

9. Verificación de Configuración Segura (Lynis, Docker Bench for Security)

Lynis (host Linux):

sudo lynis audit system --quick --report-file /tmp/lynis-report.dat

Revisar: hardening del kernel, permisos, autenticación, logging, firewall, auditoría.

Docker Bench for Security:

```
docker run -it --net host --pid host --userns host --cap-add audit_control \
  -v /etc:/etc:ro -v /usr/bin/docker-containerd:/usr/bin/docker-containerd:ro \
  -v /usr/bin/containerd:/usr/bin/containerd:ro -v /var/lib:/var/lib:ro \
  -v /var/run/docker.sock:/var/run/docker.sock:ro -v /etc/passwd:/etc/passwd:ro \
  docker/docker-bench-security
```

Validar: uso de usuarios no-root en contenedores, capacidades reducidas, read-only FS, secrets de Swarm, no exponer puertos innecesarios, política de imágenes firmadas.

10. Controles Específicos por Componente

HAProxy:

- TLS 1.2/1.3, ciphers fuertes, HSTS, OCSP stapling.
- Rate limiting (stick-table, http-request deny if exceeds), protección contra bursts.
- Health-checks activos, retry/timeout configurados.

Docker Swarm:

- Secrets en lugar de variables de entorno para credenciales.
- Constraints/labels para aislar servicios críticos.

- Resource limits y políticas de reinicio.

MariaDB:

- Deshabilitar cuentas anónimas, contraseñas fuertes, `skip_name_resolve`.
- Principio de mínimo privilegio en `GRANT`.
- Cifrado en tránsito (TLS) y en reposo si aplica.

GlusterFS:

- Permisos mínimos en puntos de montaje.
- Detección de split-brain y alertas.
- Aislamiento de red (solo nodos autorizados).

11. Evidencia, Reportes y Gestión de Vulnerabilidades

Evidencia: guardar salidas (HTML/CSV) de ZAP, Nikto, Nessus/OpenVAS, Lynis y Docker Bench; logs de JMeter; capturas de dashboards (Grafana) y configuraciones relevantes.

Reporte: consolidar hallazgos con descripción, PoC, impacto, CVSS, recomendación y responsable.

Gestión: crear tickets por hallazgo, asignar prioridad (P1–P4), fechas objetivo y verificar remediación con re-pruebas.

12. Plan de Ejecución y Cronograma de Referencia

Semana 1:

- Pentesting interno (ZAP/Nikto/sqlmap) en staging.
- Primer ciclo de remediación.

Semana 2:

- Pruebas de rendimiento bajo ataque con JMeter (baseline, spike, stress, soak).
- Segunda iteración de remediaciones.

Semana 3:

- Escaneos de vulnerabilidades autenticados (OpenVAS/Nessus) y hardening (Lynis/Docker Bench).
- Cierre de hallazgos críticos/altos.

Semana 4:

- Re-pruebas de verificación y reporte final.

13. Casos de Prueba Detallados

ID	Tipo	Descripción	Herramienta	Criterio de Éxito	Evidencia

SEC-01	DAST	ZAP Baseline sobre dominio público (cabeceras, TLS, XSS reflejado)	OWASP ZAP	0 hallazgos críticos/altos	zap_baseline.html
SEC-02	DAST	ZAP Active Scan autenticado (flujo CRUD CMS)	OWASP ZAP	Sin XSS/SQLi explotables	zap_full.html
SEC-03	ENUM	Escaneo Nikto de servidor web	Nikto	Sin métodos inseguros/archivos sensibles	nikto_report.html
SEC-04	SQLi	SQLi en parámetros GET/POST/cookies	sqlmap	No explotación confirmada	sqlmap_output.txt
SEC-05	LOAD	JMeter Spike (x3 usuarios en 60s)	JMeter	Disponibilidad ≥ 90%, Error ≤ 1%	reporte/index.html

SEC-06	VULN	Escaneo autenticado de hosts	OpenVAS/Nessus	0 vulnerabilidades críticas abiertas	export.csv
SEC-07	HARDEN	Auditoría host	Lynis	Score ≥ 70 y sin hallazgos críticos	lynis-report.dat
SEC-08	HARDEN	Auditoría contenedores	Docker Bench	Cumplimiento CIS crítico	report.json

14. Apéndice – Cabeceras y Configuraciones Recomendadas

Cabeceras HTTP (NGINX ejemplo):

```
add_header X-Frame-Options SAMEORIGIN always;
add_header X-Content-Type-Options nosniff always;
add_header Referrer-Policy no-referrer-when-downgrade always;
add_header Content-Security-Policy "default-src 'self' 'unsafe-inline' 'unsafe-eval' https: data:;" always;
add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" always;
```

HAProxy – Rate limiting (stick-table):

```
backend web
  stick-table type ip size 1m expire 10m store gpc0,conn_rate(10s)
```

```
tcp-request content track-sc0 src
acl abuse sc0_conn_rate gt 100
http-request deny if abuse
```